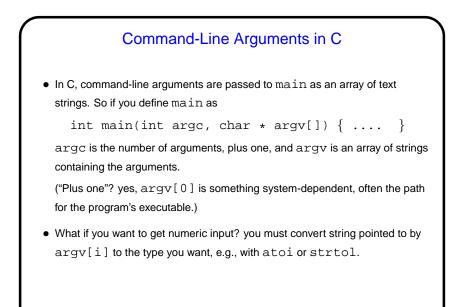


Another Way to Get Input - Command-Line Arguments

• Now that we know about arrays, pointers, and text strings, we can talk about command-line arguments. What are they? text that comes after the name of the program on the command line (e.g., when you write gcc -Wall myprogram.c, there are are two command-line arguments), possibly modified by the shell (e.g., for filename wildcards).

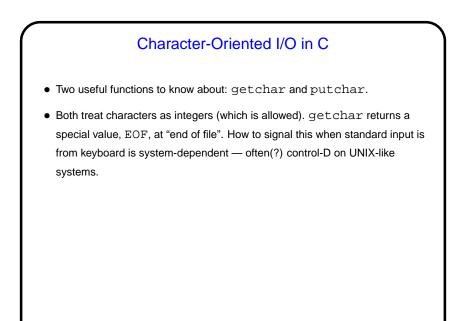
Slide 2

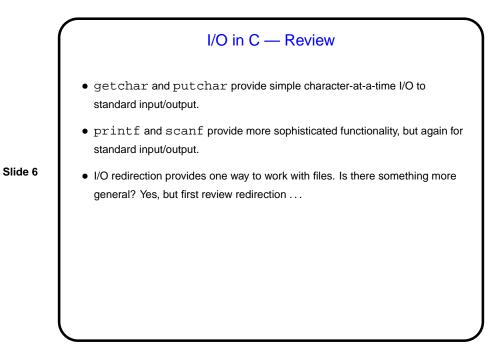
• Most programming languages provide a way to access this text, often via some sort of argument to the main function/method.

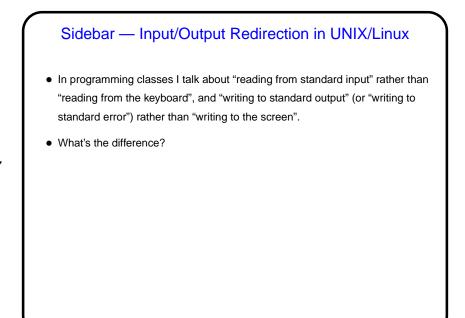


Command-Line Arguments and UNIX Shells

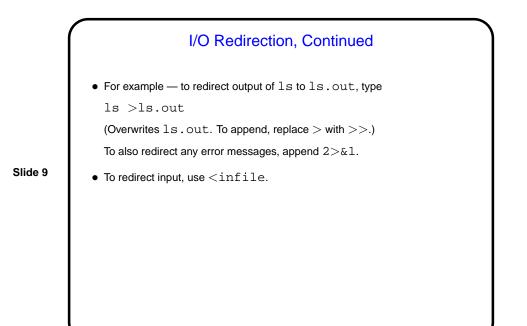
- Be aware that most UNIX shells do some preliminary parsing and conversion of what you type e.g., splitting it up into "words", expanding wildcards, etc., etc.
- If you don't want that enclose in quotation marks or use escape character (backslash).







I/O Redirection, Continued stdin (standard input) can come from keyboard, file, or from another program or shell script. stdout and stderr (standard output, error) can go to terminal or file (overwrite or append), separately or together.

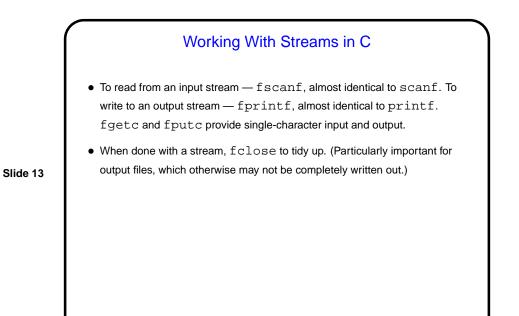


File I/O — Streams C's notion of file I/O is based on the notion of a *stream* — a sequence of characters/bytes. Streams can be *text* (characters arranged into lines separated by something platform-dependent) or *binary* (any kind of bytes). UNIX/Linux doesn't make a distinction, but some other operating systems do. Slide 10 An input stream is a sequence of characters/bytes coming into your program (think of characters being typed at the console). An output stream is a sequence of characters/bytes produced by your program (think of characters being printed to the screen, including special characters such as the one for going to the next line).

Streams in C
In C, streams are represented by the type FILE * - i.e., a pointer to a FILE, which is something defined in stdio.h.
A few streams are predefined - stdin for standard input, stdout for standard output, stderr) for standard error (also output, but distinct from stdout so you can separate normal output from error messages if you want to).
To create other streams ...

Creating Streams in C
To create a stream connected with a file — fopen.
Parameters, from its man page:

First parameter is the name of the file, as a C string.
Second parameter is how we want to access the file – read or write, overwrite or append — plus a b for binary files, also a string.
Return value is a FILE * — a somewhat mysterious thing, but one we can pass to other functions. If NULL, the open did not succeed. (Can you think of reasons this might happen?)



Reading Text Strings
Getting text-string input is surprisingly tricky. scanf (or fscanf) seems like an obvious choice, but:

it can't read a string that includes blanks, and
it has no nice way to limit the number of characters read to the size of the array being read into.

14

Getting a whole line is probably better. gets() is an obvious/simple choice for reading from standard input, but it also has no way to limit how much is read. fgets() is better. (Look at its man page.)
(Also notice puts() — simple way to write out a text string.)

(Why do you care about limiting how much is read? not doing so can crash your program or even represent a security risk ...)

